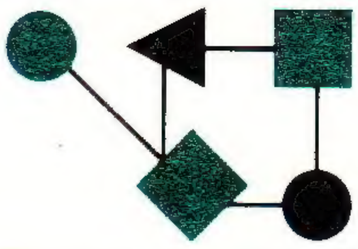
CONNEXIONS



The Interoperability Report

August 1987 Volume 1, No. 4

ConneXions - The Interoperability Report tracks current and emerging standards and technologies within the computer and communications industry.

In this issue:

ConneXions is published by Advanced Computing Environments, Inc., 480 San Antonio Road, Suite 100, Mountain View CA 94040, USA. Phone: 415-941-3399

ISSN 0894-5926

From the Editor

This month we bring you an article by Gary Krall and Keith McCloghrie of ACC on the use of transparent gateways in the Internet. Such devices offer one way to deal with the explosive growth of the network by having Ethernets appear as logical subnets on the DDN-X.25 network.

When TCP/IP is run over different kinds of physical networks, some kind of encapsulation of IP packets is necessary. This idea is explored in an article by John Romkey.

In our July issue we brought Part One of "The Internet Protocols - A Functional Overview." In this issue, Barry Shein concludes his description with a **C** programming example. Part Two is called "Anatomy of an Application Protocol."

We are only a few weeks away from the ISO Development Seminar August 31 - September 2, 1987 in Monterey, California. This seminar will give a thorough analysis of a completed implementation of the ISO protocol suite. Attendees will receive a layer-by-layer explanation and evaluation of the ISO protocol suite. Speakers include leading computer scientists Lawrence Landweber, Marvin Solomon, Rob Hagens, Nancy Hall and Sue Lebeck from the University of Wisconsin, who have worked for over two years on IBM sponsored ISO implementation projects. Also speaking will be ISO Development Expert, Marshall Rose, from Northrop Research and Technology Center. For more information on these and other upcoming events, including program and registration, contact us at 408-996-2042.

See you in Monterey!

The Use of Transparent Gateways in the Internet

by Gary Krall & Keith McCloghrie,
Advanced Computer Communications

Introduction

This article presents a new and innovative approach to attaching a TCP/IP-based 802.3 LAN to the Internet via a DDN-style X.25 network. This article is particularly topical in light of the recent availability of the ULANA Request for Proposal and other RFPs, which require inter-connectivity of LANs to the DDN via gateways.

Until now, the only approach to such an interface has been the use of an IP-or EGP-based gateway. This new approach is to use an IP-based Front End (also called a "Transparent Gateway") in place of an IP gateway. A Transparent Gateway does not require any new protocols, nor does it require any changes to the protocol implementations in any host or for that matter any other IP gateway. Instead, it considers the Ethernet as a logical (implicit) subnet of the DDN-X.25 network.

The addressing capabilities of DDN-X.25, as well as the internet address itself, allow each Ethernet host to be assigned a logical network address on the single physical X.25 connection. As a result, a Transparent Gateway can perform the necessary routing through acting, at IP-level, as a host rather than as an IP (EGP) gateway.

Limitations of IP gateways

One of the largest problems facing the current DDN-based Internet is its explosive growth. It has been estimated that there are 30,000 hosts and more than 190 active networks in today's Internet, and that these numbers are increasing at 15% per month.

The network research community has solved the potential problems associated with the explosive increase in hosts through use of the Domain System. However, the problems associated with the growth of active networks, i.e. networks with a unique network number, are still under study. This growth in network numbers presents two problems: one, a very large table size is required in each IP gateway to hold routing information for each other network number; two, the current IP gateway protocols and algorithms are approaching the limit of their capabilities.

A characteristic of the current protocols used by IP gateways is that two types of information are exchanged: one, *"Are-You-There"* and *"I-Heard-You"* messages; two, routing table updates. These are exchanged on a timer-driven basis, whether or not any information has changed. The routing table updates contain information on many networks, whether or not these networks are current destinations of traffic from the Ethernet.

Thus, an area of current research is how to change the current protocols and algorithms to cope with the increase in Network numbers, and how to avoid the network traffic load occupied by this periodic exchange of tables and status messages.

These problems show up in two ways: one, hosts lose connectivity to distant networks unless there is room in the routing tables for the source, the destination, and all intermediate networks; two, the throughput rates of the IP gateways decrease due to the extra traffic being carried on the bandwidth-limited X.25 connection, and the extra processing required to process the additional routing information.

LAN/WAN disparity

One of the problems specific to an interface between a LAN and a WAN is the wide disparity in bandwidth of the two subnetwork technologies. Congestion control is an important factor when data enters the interface at 10Mbs but can only exit at 56Kbs. Whereas large amounts of buffer space can help by smoothing the effects of receiving bursts of traffic, RFC 970 explains that large amounts of storage do not solve the problem. RFC 970 recommends the separating of streams according to source host in order to equalize the effects of congestion control.

The network research community is urging the use of subnetting (see RFC 950) wherever possible in order to slow the rate of growth of network numbers. However, RFC 950's form of subnetting only allows a reduction in network numbers when there are multiple co-located LANs. It does not, in general, provide a mechanism to avoid having an additional network number for an isolated LAN.

A less pressing area of study is to provide the ability to load-level between two IP gateways. Existing protocols provide for multiple IP gateways to be used between two networks. However, existing support only allows for redundancy, not for load-leveling. Thus, if two IP gateways were used between an Ethernet and an X.25 network, all traffic would probably flow through one of them, with the other sitting idle.

Advantages of Transparent Gateways

A Transparent Gateway provides better performance since there is less overhead traffic on the congested X.25 link. For example, a Transparent Gateway requires no exchange (i.e. both directions) of periodic routing table updates and status messages. It only requires routing information to be received for destinations currently being used, and only when routing changes, as is the case for directly attached hosts.

Transparent Gateways have the ability to provide flow control on a source-host/destination-host pair basis rather than on a destination-host-only basis. Separate X.25 virtual circuits are established for each X.25 logical address assigned to one or more of the Ethernet hosts. Thus, the use of multiple logical addresses provides the additional flow control rather than having all packets lumped into a single virtual circuit as would be established by an IP gateway. Thus, the few packets of an interactive connection from one Ethernet host would not be delayed by the many packets of a bulk transfer from another Ethernet host.

Transparent Gateways do not implement any of the current gateway protocols, which are currently under review and are subject to change (see RFC 1009 and other proposed changes such as "Son-of-EGP").

continued on next page

Use of Transparent Gateways *(continued)*

Transparent Gateways ease the problems associated with the increasing number of active networks by not requiring an extra network number for an isolated Ethernet.

Transparent Gateways will also make it less likely for hosts on the Ethernet to suffer loss of connectivity to distant networks due to lack of routing table space, since space will not be required for a network number for the Ethernet.

Finally, multiple Transparent Gateways between an Ethernet and a DDN-X.25 network can load-level the traffic being exchanged between the two networks. This load-leveling occurs as a result of the necessary virtual circuits between Ethernet hosts and X.25 hosts being spread across the multiple Transparent Gateways.

Functional overview of Transparent Gateways

As the name implies, Transparent Gateways are "transparent" to the hosts they are serving; that is, hosts communicate using the same protocols whether they are communicating via a Transparent Gateway or with any other host on the Ethernet (i.e., not through the gateway); these protocols are ARP, IP and any transport-layer protocol they choose (e.g., TCP). The advantages of this transparency are that not only can different hosts be supported irrespective of host's type, but also no development is required in the host provided that:

a) The host's TCP/IP implementation conforms to the latest protocol standard (as specified in the RFCs), and

b) All hosts have an implementation of the Address Resolution Protocol (ARP).

Each Transparent Gateway has its own Internet Address; this is comprised of the network number of the DDN-X.25 network and the DDN-X.25 physical address of the gateway's physical connection to the PSN.

Address usage by Transparent Gateways

Transparent Gateways can support two types of addressing schemes: physical and logical.

Physical Addressing -- Physical addressing is limited to Class A networks and uses the **n.h.l.i** format. In this mode, the internet addresses of hosts on the Ethernet are formed as follows:

The **n** field is the network number of the DDN-X.25 network to which a Transparent Gateway is connected (e.g., for MILNET, the network number is 26).

The **h** field is the number of the PSN's port to which a Transparent Gateway is connected.

The **l** field is configured as a unique number for each host on the Ethernet.

The **i** field is the number of the PSN to which a Transparent Gateway is connected.

Using physical addressing, the internet addresses of the hosts on the Ethernet are identical except for the **l** field. For example, if three hosts and a Transparent Gateway are on an Ethernet with the gateway connected to port #3 on PSN #2, and the PSN connected to network #26, then the internet addresses for the hosts are 26.3.1.2, 26.3.2.2, and 26.3.3.2. The internet address of the gateway is 26.3.0.2.

Logical Addressing -- When multiple Transparent Gateways are used to connect the LAN to the DDN-X.25 network for load-leveling and redundancy, logical addressing must be used.

Logical addressing allows multiple logical addresses to be enabled on a single physical PSN port, as well as allowing a single logical address to be enabled on multiple physical ports.

Logical addresses themselves are derived from a five-digit host identifier which has been assigned by the network administration and is used for addressing the Ethernet hosts that the Transparent Gateway is "front-ending." The PSN port is then "enabled" by the Network Operations Center (NOC) with these identifiers to allow for the ability for logical addresses to be mapped into physical addresses by the PSN. (For a more detailed discussion see RFC 1005, "The ARPANET AHIP-E Host Access Protocol.")

Logical addresses can be used on Class A or Class B networks. For Class A, the internet address is formed as follows :

The **n** field is the network number of the DDN-X.25 network to which the Transparent Gateway is connected.

The **h** field is the quotient of the host identifier divided by 256.

The **l** field in a Class A logical address is a unique number for each host on the Ethernet which shares this logical address. If only one Ethernet host uses this logical address, then this field can be set to zero. If many Ethernet hosts share the X.25 Logical address, then each of these hosts must be assigned a unique value.

The **i** field is the remainder from the division of the host identifier by 256.

For example, the logical address of a Class A host with the host identifier 33537 is 26.131.0.1. (33537 divided by 256 is 131 [the **h** field] with 1 remaining [the **i** field]. 26 is the network number.)

Implementation of a Transparent Gateway

The ACS 4020, designed and developed by ACC, is an example of a Transparent Gateway. Presented below is an overview of how routing decisions are made and how flow control between the two network types is facilitated. In addition, a discussion is also presented on the use of multiple ACS 4020s to provide load-leveling and redundancy. We conclude with an operational overview within an Internet environment.

continued on next page

Use of Transparent Gateways *(continued)*

Routing

The ACS 4020 maintains two internal routing tables: its ARP Table, and its Gateway Routing Table. The ARP Table contains the current internet-address to Ethernet-address equivalences for hosts on the Ethernet. It is updated through the use of ARP requests/replies. Entries in the ARP Table are aged, to cause new ARP requests to be generated periodically for each known equivalence.

The internal Gateway Routing Table contains an entry for each IP network number other than the network number of the local DDN-X.25 network. An entry contains the address of the current gateway on the local DDN-X.25 network to use for the entry's network number. At system-restart the table is empty. A new entry is created whenever there is a packet, received from the Ethernet, to be routed to a network number which is not currently in the table. New entries are initialized with the current-default-gateway address. The current-default-gateway address is initially set as the first of the default gateways; it is modified to the next in the list (cyclically) if its current value is ever reported to be unreachable by the DDN-X.25 network. Existing entries in the Gateway Routing Table are updated whenever an Internet Control Message Protocol (ICMP) "Redirect" message is received from the DDN-X.25 network, or if any entry in the table contains a gateway address which is reported as unreachable by DDN-X.25.

For each packet received from the Ethernet, if it is an ARP request, an ARP reply is sent providing the internet address is not in the second set defined above. If it is an IP packet, the destination internet address is examined to determine the DDN-X.25 address to which it is to be sent. If the destination internet address is an IP level broadcast address, the packet is discarded. If the destination internet address is on the local DDN-X.25 regional network, then the DDN-X.25 address is calculated directly; otherwise, the DDN-X.25 address of an IP-gateway on the local DDN-X.25 network is determined by examining the Gateway Routing Table (see above). Then the packet is transmitted to the PSN, using the determined DDN-X.25 address as its local network destination.

Any packets received from the DDN-X.25 network which are not IP packets are discarded. For IP packets, the IP header is examined to ensure the checksum, the version number and the length are valid. If valid, the destination internet address is extracted. For any type of internet address other than A, B, or C the packet is discarded.

Otherwise, the destination internet address is extracted from the IP header. If there is already an entry in the ARP table for this internet-address, the entry's Ethernet address is used as the destination address on the Ethernet. If no ARP table entry exists, an ARP request is sent out for the internet-address, and the packet is queued awaiting an ARP reply.

Each packet received from the DDN-X.25 network is examined to determine if it is an ICMP Redirect message. If so, it is used to update the appropriate entry in the Gateway Routing Table providing it was received from a "known" gateway; "known" gateways are those currently in the routing table, plus the set of default gateways.

Flow control

Data transmission between the ACS 4020 and the DDN-X.25 network is flow-controlled at both levels 2 and 3 of the DDN-X.25 protocol. In contrast, there is no flow control on transmissions between the ACS 4020 and hosts on the Ethernet. This, of course, is typical of an Ethernet environment and of the datagram concept of an IP network, where guaranteed ordered delivery requires the use of an end-to-end transport protocol (e.g., TCP). Some assurance, however, is needed that:

a) Ethernet hosts do not repeatedly overrun the ACS 4020 with more transmissions than the DDN-X.25 network can absorb, and

b) There is a limit to how much Ethernet traffic the ACS 4020 can send in one burst to a host.

These assurances are provided by a large amount of buffer space in the ACS 4020, and by flow control in the host's level 4 (and higher) protocols. For example, if TCP is used for the majority of transmissions, then the end-to-end flow control on TCP connections provides a window size specifying the maximum amount of data to be in transmission on each connection. This window size is dynamically adjusted by the receiving TCP implementation according to its current buffer space limitations.

In the ACS 4020 to-host direction, the host's assurance of being able to take the maximum burst of traffic is that the size of such a burst is limited by the aggregate of all window sizes, and that the host itself controls these.

In the other direction, the provision of 1.5Mbs of buffer space assures that the ACS 4020 is not repeatedly overrun. With this amount of memory, the capacity to buffer data which the DDN-X.25 network cannot yet absorb is estimated at 400 TCP connections (assuming an average window size of 1Kbs).

Handling bad TCP implementations

Nevertheless, congestion can cause TCP retransmission timers to expire, resulting in additional IP datagrams being received by the ACS 4020. Correctly implemented TCPs will dynamically respond to this by slowing their rate of retransmission. However, some means of limiting the number of queued datagrams is required to cater to "rogue" TCP implementations. This means is provided for by the use of the "Time to Live" field in the IP header. As specified in the IP specification this field must be decremented every second it is queued in the ACS 4020, with the datagram being discarded if this field reaches zero.

As mentioned above, data transmission between the ACS 4020 and the X.25 network is flow controlled at both levels 2 and 3. Note that flow control at X.25 level 3 provides independent flow control for each X.25 logical address. Thus, independent flow control for individual Ethernet hosts is obtained if a unique X.25 logical address is assigned for each. This would prevent one "rogue" TCP implementation from starving all the others of all service.

continued on next page

Use of Transparent Gateways *(continued)*

Load-leveling and redundancy

For maximum redundancy and dynamic load-leveling, multiple ACS 4020s are needed. They should be configured with identical sets of internet addresses.

On the DDN-X.25 side, redundancy and load-leveling are both achieved through DDN-X.25's Logical- Addressing. On the Ethernet side, multiple 4020s respond to ARP requests specifying internet addresses of hosts not on the Ethernet. The ARP protocol specifies that the latest of received ARP replies be used for establishing address equivalences.

Redundancy is achieved because one ACS 4020 can fulfill the whole functionality (at a lower aggregate throughput) if all other 4020s were to fail.

Load-leveling is achieved on the Ethernet side, dynamically (i.e. non-deterministically) because in some cases, one ACS 4020's ARP reply will be the last to arrive (at the requesting host), and in other cases, another 4020's ARP reply will be the last to arrive. With sufficient hosts and destinations, the multiple ACS 4020s will equally share the load.

Each ACS 4020 will delay its ARP reply by an amount inversely proportional to the number of packets it has processed over the past few seconds/minutes. Therefore, the least busy 4020 will be the most likely to have its ARP reply arrive last at the requesting host.

Operational example

This example assumes an environment as depicted by Figure 1, with the following assignment of addresses:

- Internet addresses "A" through "L" are hosts on the Ethernet,
- Internet address "X" is a host on the local DDN-X.25 network,
- Internet addresses "G1" and "G2" are gateways on the local DDN-X.25 network, and
- Internet address "Y" is a host on a non-local network "N".

A normal configuration for the ACS 4020 would be as follows; the first set: A through L; the second set: A through L; the third set: G1 and G2. (We are assuming that there is an alternate route between N1 and N2 via one or more gateways which are not shown.)

Suppose host A must send a packet to host Y. Host A would issue an ARP request either for host Y's internet address, or for a gateway (e.g., G1's) internet address. Since neither Y nor G1 are in the second set of addresses, the ACS 4020 would generate an ARP reply. Host A can now send its packet with the 4020's Ethernet-address as its destination. When it arrives at the ACS 4020, the packet's IP header is examined to discover the destination address is Y.

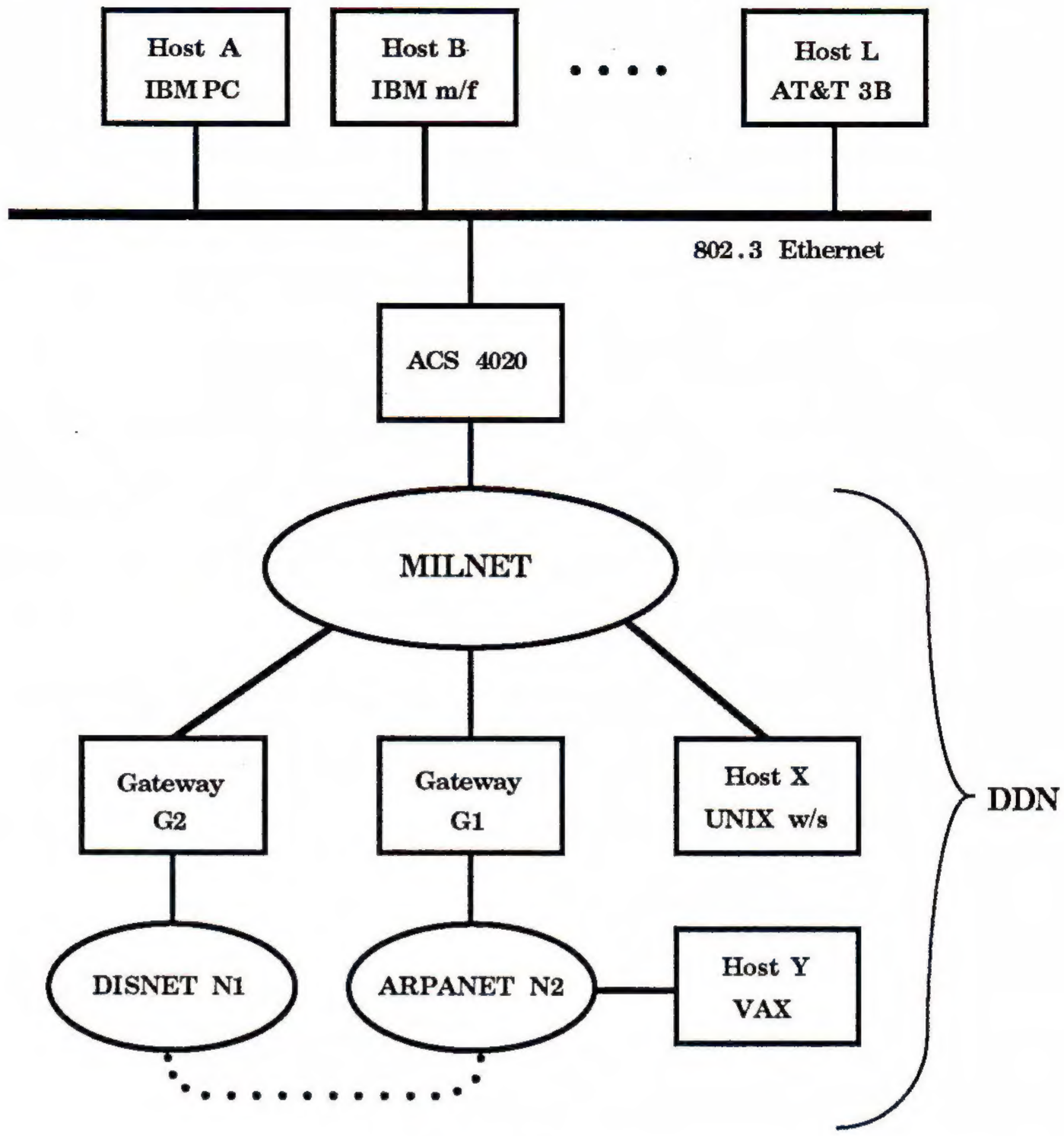


Figure 1

Since host Y is not on the local DDN-X.25 network, the packet cannot be sent directly (as would be the case if the destination were host X). A Gateway Routing Table entry would be created for network N2 and initialized with gateway address G1, and a DDN-X.25 virtual circuit requested to G1. If G1 were down, the virtual circuit would not be established, and the Gateway Routing Table entry would be updated to contain G2 (the next gateway in the third set). Otherwise, the packet would be sent to G1. G1 would forward the packet to G2 (the correct gateway for delivery to Y) and would return an ICMP Redirect message to the source internet address. On arrival at the ACS 4020, this packet would be detected as a Redirect and the Gateway Routing Table updated to contain G2's address as the gateway of choice. Subsequent packets sent by A would not require an ARP request, and the ACS 4020 would send them directly to G2.

A packet received from the DDN-X.25 PSN addressed to host A would cause the ACS 4020 to issue an ARP request for internet address A. When host A replies with an ARP reply, the packet would be forwarded using host A's Ethernet address as its destination.

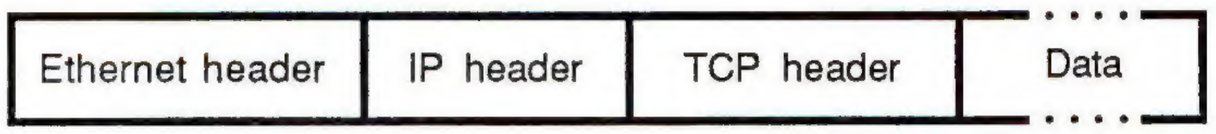
It has been shown in a number of customer installations that transparent gateways are a viable alternative to IP EGP-based gateways. They are ideally suited for providing Internet access without the need for complex routing protocols to sites which have a relatively small number of hosts connected via a LAN. Devices such as the ACS 4020 do not contribute to the exponential growth of network numbers which continue to hamper effective management and performance of the Internet.

Trailers or No Trailers

by John Romkey

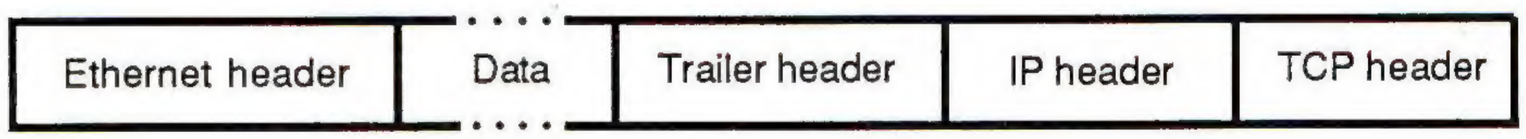
Running TCP/IP over a LAN

For each type of network hardware you might want to run IP over, an *encapsulation* for IP on that hardware must be defined. There are several issues involved in defining the encapsulation. First, there are physical issues involving how the IP packet is arranged in the hardware packet, and how it is identified as an IP packet. Second, there must be an algorithm for translating between IP and hardware addresses (if this network hardware even *has* addresses). The encapsulations for IP on the most common IP-supporting networks (Ethernet, Arpanet, IEEE 802.5 Token Ring) are all specified in RFC's. There is a particular problem with the way a number of IP-implementations encapsulate IP on Ethernet, though.



IP over Ethernet

The *standard* encapsulation of TCP/IP on 10Mbps Ethernet (IP was also run on an earlier, 3Mbps version of Ethernet) was specified in RFC 894, after several years of running IP on Ethernet. Berkeley 4.2 UNIX introduced an alternate method of encapsulating IP that was incompatible with the old one, but which was much more efficient to use on VAX processors. (The alternate encapsulation allowed packets to be assembled by the UNIX kernel without copying them, whereas the standard format required copying portions of the data). The standard format was to start the packet with an Ethernet header, immediately followed by the IP header, followed by the UDP or TCP header, followed by the UDP or TCP data. Berkeley UNIX would usually use this format, but if the data was a multiple of 512 bytes long (the page size of 4BSD on a VAX is 512 bytes), it would use its own *trailer* format. A trailer packet has the UDP or TCP data immediately after the Ethernet header, followed by the IP and UDP or TCP headers.



Unix vs. non-Unix machines

Using trailers to communicate between Berkeley UNIX machines is a reasonable idea; they increase throughput substantially on some computer hardware. Many common implementations of TCP/IP are derived from Berkeley UNIX: DEC's Ultrix, the Sun Microsystems operating system and Wollongong's VMS TCP/IP are a few examples. And, of course, many people run Berkeley UNIX out of the box. The popularity of Berkeley UNIX has greatly increased the number of computers running TCP/IP, and has made TCP/IP much more visible to the rest of the world. But not all computers run Berkeley UNIX.

When a computer that is *not* running 4.2 receives a trailer packet, it's all over for its network connection. The TCP on the 4.2 side will keep resending the trailer packet over and over again, and never get a response, because the other computer won't recognize the trailer packet as a different representation of IP over Ethernet.

A common occurrence is for a user to telnet from a non-4.2 system to a 4.2 system and do something to generate a lot of output (perhaps display a file) and the telnet connection will hang until the non-4.2 system or the user gives up in despair. A *few* non-4.2 TCP/IP implementations will recognize incoming trailers, but these TCP/IP's are exceptions rather than the rule.

Turning off trailers

By default, 4.2 UNIX and many of the systems derived from it have trailers enabled, but there is a way to disable them on a per-interface basis. The command /etc/ifconfig controls the behavior of network interfaces under UNIX. It takes an option, -trailers, that tells it to disable trailers. To disable them on interface *il0*, you would issue the command:

/etc/ifconfig il0 -trailers

Just typing the command with the interface name but no options will display the current status of that interface. Refer to your UNIX manual for more information.

A word of advice to system administrators: if you have any non-4.2-derived systems at your site, turn off trailers on all your 4.2-derived systems. It will save you the hassle of trying to figure out why your non-4.2 systems don't work right with your 4.2 systems, and you probably won't notice the change in performance.

A word of advice to vendors: if your system doesn't currently understand trailers, think about allowing it to receive them but not ever send them. The trailer format is rather simple, and is documented in RFC 893. Accepting incoming trailers will save you the support work for those sites which haven't disabled them. If you really want to be able to send trailers, implement the 4.3 UNIX trailer negotiation mechanism.

4.3BSD and trailer negotiation

4.3 UNIX still uses trailers, but doesn't have the problem of sending trailers to machines which don't understand them. The fix is rather simple. Trailers are represented on Ethernet as a different Ethernet packet type from the standard IP encapsulation (after all, the computers have to be able to tell them apart). When 4.3 goes through the stage of translating IP addresses to Ethernet addresses, it invokes the Address Resolution Protocol (ARP - see RFC 826). An ARP address translation request includes information about the protocol the address belongs to. 4.3 simply uses the trailer type field instead of the normal IP type field on its first attempt to translate an address. Other 4.3 systems know to reply to this. If it gets a reply, the 4.3 system remembers that it's safe to use trailers with the other system. If it doesn't get a reply, it tries the standard IP type field and remembers later *not* to use trailers.

JOHN ROMKEY received his B.S. in Computer Science from MIT in 1985. While there, he worked with Prof. Jerome Saltzer and Dr. David Clark for three and a half years on the PC/IP project, a popular public domain implementation of TCP/IP for IBM PC's. After graduation, he spent a year as a staff member with Dr. Clark, and moved on to help found FTP Software, Inc., where he was Director of Software Development until July 1987. Although he retains ties with FTP, he currently spends his time consulting, developing new network services for IBM PC's, and trying to write science fiction.

The Internet Protocols -- A Functional Overview Part Two: Anatomy of an Application Protocol

by Barry Shein, Boston University

Following the protocol overview in the July issue of *ConneXions*, we present a C programming example based upon the 4.2 UNIX operating system developed at the University of California, Berkeley. It is a very simple implementation of a whois client and server [RFC 954] which allows someone to look up information about a user on another system using a TCP stream. The example code is also available via anonymous FTP from BU-CS.BU.EDU in the file pub/Connexions.example.

In general, applications will contain a server and one or more clients. The server sits and waits for an incoming request. The clients are typically started when a user requests some service. UNIX 4.2, like most systems, provides TCP/IP within the operating system itself so we will only have to specify which machine we need to speak with and our request once the connection has been established.

WHOIS client

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
/*
 * WHOIS client program
 *     Usage: whois host username
 */
main(argc,argv)
int argc;
char **argv;
{
  int s;
  int i;
  struct sockaddr_in sa;
  struct hostent *hp;
  struct servent *sp;
  char buf[BUFSIZ];
  char *myname;
  char *host;
  char *user;

  myname = argv[0];
/*
 * Check that there are two command line arguments
 */
if(argc != 3) {
  fprintf(stderr,"Usage: %s host username\n",myname);
  exit(1);
}
host = argv[1];
user = argv[2];
/*
 * Look up the specified hostname
 */
if((hp = gethostbyname(host)) == NULL) {
  fprintf(stderr,"%s: %s: no such host?\n",myname,host);
  exit(1);
}
```

```
/*
 * Put host's address and address type into socket structure
 */
bcopy((char *)hp->h_addr,(char *)&sa.sin_addr,hp->h_length);
sa.sin_family = hp->h_addrtype;
/*
 * Look up the socket number for the WHOIS service
 */
if((sp = getservbyname("whois","tcp")) == NULL) {
  fprintf(stderr,"%s: No whois svc on this host\n",myname);
  exit(1);
}
/*
 * Put the whois socket number
 * into the socket structure.
 */
sa.sin_port = sp->s_port;
/*
 * Allocate an open socket
 */
if((s = socket(hp->h_addrtype,SOCK_STREAM,0)) < 0) {
  perror("socket");
  exit(1);
}
/*
 * Connect to the remote server
 */
if(connect(s,&sa,sizeof sa) < 0) {
  perror("connect");
  exit(1);
}
/*
 * Send the request
 */
if(write(s,user,strlen(user)) != strlen(user)) {
  fprintf(stderr,"%s: write error\n",myname);
  exit(1);
}
/*
 * Read the reply and put to user's output
 */
while((i = read(s,buf,BUFSIZ)) > 0)
  write(0,buf,i);
close(s);
exit(0);
}
```

WHOIS server

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#include <pwd.h>
/*
 * WHOIS server
 *   Listens on the well-known WHOIS port (43),
 *   requires super-user privilege to run.
 */

/*
 * How many connections we are willing to fall behind
 */
#define BACKLOG         5
```

continued on next page

The Internet Protocols -- A Functional Overview *(continued)*

```
/*
 * Longest host name, arbitrary
 */
#define MAXHOSTNAME     32

main(argc,argv)
int argc;
char **argv;
{
  int s, t;
int i;
struct sockaddr_in sa, isa;
struct hostent *hp;
char *myname;
struct servent *sp;
char localhost[MAXHOSTNAME+1];

myname = *argv;
/*
 * Look up the WHOIS service entry
 */
if((sp = getservbyname("whois","tcp")) == NULL) {
  fprintf(stderr,"%s: No whois svc on this host\n",myname);
  exit(1);
}
/*
 * Get our own host information
 */
gethostname(localhost,MAXHOSTNAME);
if((hp = gethostbyname(localhost)) == NULL) {
  fprintf(stderr,"%s: can't get local host info?\n",myname);
  exit(1);
}
/*
 * Put the WHOIS socket number and our address info
 * into the socket structure
 */
sa.sin_port = sp->s_port;
bcopy((char *)hp->h_addr,(char *)&sa.sin_addr,hp->h_length);
sa.sin_family = hp->h_addrtype;
/*
 * Allocate an open socket for incoming connections
 */
if((s = socket(hp->h_addrtype,SOCK_STREAM,0)) < 0) {
  perror("socket");
  exit(1);
}
/*
 * Bind the socket to the service port
 * so we hear incoming connections
 */
if(bind(s,&sa,sizeof sa,0) < 0) {
  perror("bind");
  exit(1);
}
/*
 * Set maximum connections we will fall behind
 */
listen(s,BACKLOG);
```

```
/*
 * Go into an infinite loop waiting for new connections
 */
for(;;) {
  i = sizeof isa;
  /*
   * We hang in accept() while waiting for new customers
   */
  if((t = accept(s,&isa,&i)) < 0) {
    perror("accept");
    exit(1);
  }
    whois(t);       /* perform the actual WHOIS service */
    close(t);
  }
}

/*
 * Get WHOIS request from remote host and format a reply.
 */
whois(sock)
int sock;
{
  struct passwd *p;
  char buf[BUFSIZ];
  int i;

  /*
   * Get one line request
   */
  if((i = read(sock,buf,BUFSIZ)) <= 0)
    return;
  buf[i] = '\0';  /* Null terminate */
  /*
   * Look up the requested user
   * and format reply
   */
  if((p = getpwnam(buf)) == NULL)
    strcpy(buf,"User not found\n");
  else
    sprintf(buf,"%s: %s\n",p->pw_name,p->pw_gecos);
  /*
   * Return reply
   */
  write(sock,buf,strlen(buf));
  return;
}
```

CONNEXIONS
480 San Antonio Road
Suite 100
Mountain View, CA 94040

FIRST CLASS MAIL
U.S. POSTAGE
P A I D
SAN JOSE, CA
PERMIT NO. 1

CONNEXIONS

PUBLISHER Daniel C. Lynch

EDITOR Ole J. Jacobsen